1. OpenCV의 채널 처리 함수에 대해서 아는 대로 기술하시오.

|  |
| --- |
| cv2.merge() : 단일채널 배열을 다채널 배열로 합성하는 함수  cv2.split() : 다채널 배열을 단일채널 배열로 분리하는 함수 |

2. OpenCV의 사칙 연산을 수행하는 함수와 연산의 수행 방법에 대해서 기술하시오.

|  |
| --- |
| cv2.add(), cv2.subtract(), cv2.multiply(), cv2.divide(), cv2.addWeighted()  입력인수는 넘파이의 ndarray 객체끼리만 가능하다. |

3. 행렬(ndarray)을 초기화하는 방법들에 대해서 기술하고, 각 방법으로 선언하시오.

|  |
| --- |
| np.zeros() : 0원소 행렬 선언  np.ones() : 1원소 행렬 선언  np.full() : 임의의 수로 행렬 선언 |

4. 사칙연산이나 논리 비트 연산에서 마스킹(masking)을 사용할 수 있다. 마스크 행렬에 대한 의미와 사용법에 대해서 설명하시오.

|  |
| --- |
| 원하는 위치만을 연산대상으로 지정하고자 할 때 마스킹을 사용한다.  mask=(연산하고자 하는 위치)로 지정하여 마스킹을 사용할 수 있다. |

5. cv2.reduce() 함수에 대해서 설명하고, 특히 축소 시에 사용하는 연산 옵션에 대해서 상세히 설명하시오.

|  |
| --- |
| cv2.reduce(src, dim, rtype, [,dst[,dtype]]) : 행렬을 열, 행방향으로 축소시킨다.  -dim  dim이 0이면 열 방향으로 연산>>1행으로 축소  dim이 1이면 행 방향으로 연산>>1열으로 축소  -rtype  cv2.REDUCE\_SUM : 행렬의 모든 행들을 합한다.  cv2.REDUCE\_AVG : 행렬의 모든 행들을 평균한다.  cv2.REDUCE\_MAX : 행렬의 모든 행들의 최댓값을 구한다.  cv2.REDUCE\_MIN : 행렬의 모든 행들의 최솟값을 구한다. |

6. 다음 예시 코드는 컴파일 에러가 발생한다. 에러가 발생하는 부분을 수정하고 실행결과를 적으시오.

|  |
| --- |
| import numpy as np, cv2  m1=np.array([1,2,3,1,2,3], np.int8) m2=np.array([3,3,4,2,2,3], np.int8)  m3=cv2.add(m1,m2) m4=cv2.subtract(m1,m2)  print("[m1]=%s"%m1) print("[m2]=%s"%m2) print("[m3]=%s"%m3.flatten()) print("[m4]=%s"%m4.flatten()) |

|  |
| --- |
| import numpy as np, cv2  data=[1,2,3,4,5,6,7,8,9,10,11,12] m1=np.array(data).reshape(2,2,3)  r, g, b=cv2.split(m1)  print("[m1] = %s"%m1) print("[r,g, b]=%s, %s, %s"%(r, g, b)) |

7.

|  |
| --- |
| import numpy as np, cv2  logo=cv2.imread("c:/computervision/chap05/image/img4.jpg", cv2.IMREAD\_COLOR) logo = logo[:600,:] if logo is None: raise Exception("영상 입력 오류")  blue, green, red = cv2.split(logo)  zero=np.zeros(logo.shape[:2], np.uint8) list=[blue, zero, zero] blue\_image=cv2.merge(list) list=[zero, green, zero] green\_image=cv2.merge(list) list=[zero, zero, red] red\_image=cv2.merge(list)  cv2.imshow('1', blue\_image) cv2.imshow('2', green\_image) cv2.imshow('3', red\_image) cv2.waitKey() |

8.

|  |
| --- |
| import numpy as np, cv2  image = cv2.imread("c:/computervision/chap05/image/img2.jpg", cv2.IMREAD\_GRAYSCALE) image=image[400:1200,400:1200] if image is None: raise Exception("영상 입력 오류")  center = (190, 170) size = (150, 200) ellipse=np.full((500,500), 0, np.uint8) cv2.ellipse(ellipse, center, size, 0, 0, 360, 255, -1)  (w,h)=ellipse.shape[:2] roi=image[:w,:h]  background=cv2.bitwise\_and(roi, roi, mask=ellipse)  cv2.imshow("image", background) cv2.waitKey() |

9.

|  |
| --- |
| import numpy as np,cv2  m=[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18] m=np.array(m, np.uint8).reshape(3, 6)  m\_reduce1 = cv2.reduce(m, 0, cv2.REDUCE\_AVG) m\_reduce2 = cv2.reduce(m, 1, cv2.REDUCE\_AVG)  print("orginal=\n%s\n" %m) print("reduce1=\n%s\n" %m\_reduce1) print("reduce2=\n%s\n" %m\_reduce2) |

10.

cv2.mean()

|  |
| --- |
| import numpy as np, cv2  capture = cv2.VideoCapture(0) if capture.isOpened()==False:  raise Exception("카메라 연결 안됨")  while True:  ret, flame = capture.read()  if not ret:break  if cv2.waitKey(30) >=0 : break   mask = np.zeros(flame.shape[:2], np.uint8)  mask[100:201, 200:401]=255  mean = cv2.mean(flame, mask)  cv2.rectangle(flame, (200,100),(400, 200), (255, 0 ,0 ), 5)  print("mean=",mean)  cv2.imshow('1', flame)  capture.release() |

원소순회방법

|  |
| --- |
| import numpy as np, cv2  capture = cv2.VideoCapture(0) if capture.isOpened() == False:  raise Exception("카메라 연결 안됨")  while True:  ret, flame = capture.read()  if not ret: break  if cv2.waitKey(30) >= 0: break   b\_sum, g\_sum, r\_sum = 0, 0, 0  for x in range(200, 401):  for y in range(100, 201):  b, g, r = cv2.split(flame)  b\_sum += b[y, x]  g\_sum += g[y, x]  r\_sum += r[y, x]  mean = [b\_sum/20000, g\_sum/20000, r\_sum/20000]   cv2.rectangle(flame, (200, 100), (400, 200), (255, 0, 0), 5)  print("mean=", mean)  cv2.imshow('1', flame)  capture.release() |

11.

|  |
| --- |
| import numpy as np ,cv2  capture=cv2.VideoCapture(0) if capture.isOpened() == False:raise Exception("카메라 연결 안됨")  while True:  ret, flame = capture.read()  if not ret:break  if cv2.waitKey(30)>=0 : break   cv2.rectangle(flame, (30, 30), (350, 270), (255, 0, 0), 5)  flame = flame[30:271, 30:351]   M = np.float32([[1, 0, 30], [0, 1, 30]])  main = cv2.warpAffine(flame, M, (400, 300))  cv2.imshow('main\_window', main)  capture.release() cv2.destroyAllWindows() |

12.

|  |
| --- |
| import numpy as np, cv2  video = cv2.VideoCapture("c:/computervision/chap05/image/mv.mp4") if video.isOpened() == False:raise Exception("동영상 연결 안됨")  while video.isOpened():  ret, flame = video.read()  if not ret : break  if cv2.waitKey(30)>=0 : break   flame1 = flame[:800, 300:1000]  flame1[100:200, 100:200] += 50   mask = np.zeros(flame1.shape[:2], np.uint8)  mask[500:600, 500:600] = 255  flame2 = cv2.convertScaleAbs(flame1, alpha=3, beta=1)   flame2 = cv2.copyTo(flame2, mask, flame1)  cv2.imshow('1', flame2)  cv2.waitKey(0) video.release() |

13.

|  |
| --- |
| import numpy as np, cv2  def print\_rects(rects):  for i, (x, y, w, h, a) in enumerate(rects):  print("rexts[%i] = [(%3d,%3d) from (%3d,%3d)] %5d" % (i, x, y, w, h, a))  rands = np.zeros((5,5), np.uint16) starts = cv2.randn(rands[:,:2], 100, 50) ends = cv2.randn(rands[:, 2:-1],300, 50)  sizes = cv2.absdiff(starts, ends) areas = sizes[:,0] \* sizes[:,1] rects = rands.copy() rects[:, 2:-1] = sizes rects[:,-1]=areas  index = cv2.sortIdx(areas, cv2.SORT\_EVERY\_COLUMN).flatten()  print\_rects(rects[index.astype('int')]) |

14.

|  |
| --- |
| import numpy as np, cv2  data = [3, 6, 3, -5, 6, 1, 2, -3, 5] m1 = np.array(data, np.float32).reshape(3,3) m2 = np.array([2, 10, 28], np.float32)  ret, inv = cv2.invert(m1, cv2.DECOMP\_LU) if ret:  \_, ans = cv2.solve(m1, m2, cv2.DECOMP\_LU)   print("answer : ", ans.flatten())   else:  print("오류") |

15.

|  |
| --- |
|  |